Biologically Inspired Computing

EECS 6180

Homework 4

Genetic Programming

Edris Amin

Mar. 15, 2012

Biologically Inspired Computing EECS6180 Edris Amin
Homework 4 Genetic Programming March 15, 2012

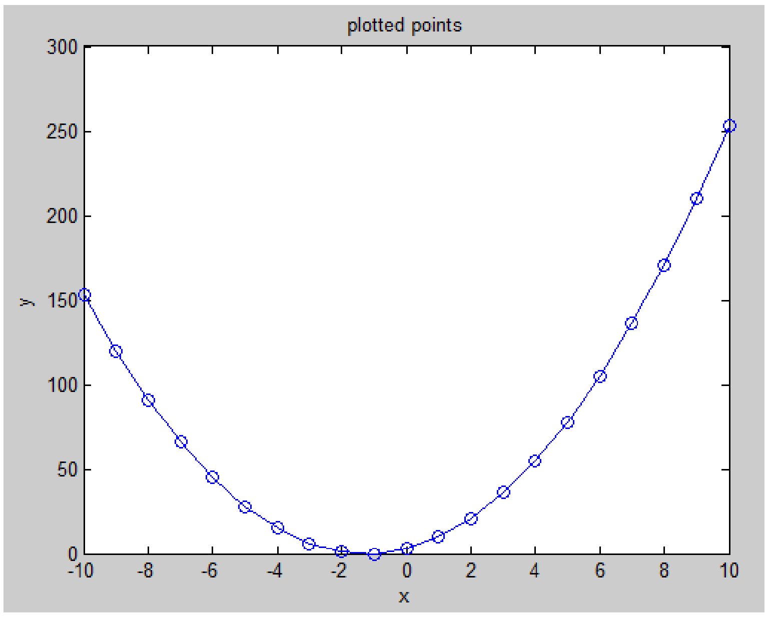
Question:

Determine using genetic programming (GP), the computer program (S-expression) that produces exactly the outputs presented in the table below for each value of x. The following hypotheses are given:

1. Use only functions with two arguments (binary trees)
2. Largest depth allowed for each tree: 4
3. Function set: F = {+, *}
4. Terminal set: T = {0, 1, 2, 3, 4, 5, x}

Table of input data

X	Y
-10	153
-9	120
-8	91
-7	66
-6	45
-5	28
-4	15
-3	6
-2	1
-1	0
0	3
1	10
2	21
3	36
4	55
5	78
6	105
7	136
8	171
9	210
10	253



Solution:

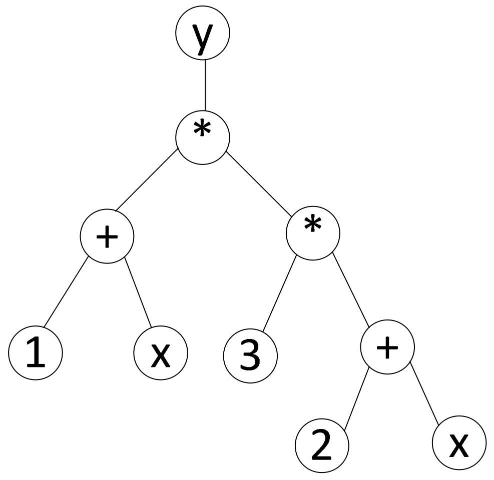
The solution for this is to evolve a tree representing an S-expression to find a function which models the relationship between x and y accurately. An example of a suitable S-expression and tree is show below.

Actual equation:

$Y = 3*(2+x)*(1+x)$

S-expression:

Y = times(times(3,plus(2,x)), plus(1,x))



GPTips, developed by Dominic Searson, is a set of MATLAB functions and scripts designed to evolve a tree of S-expressions to evaluate such problems it is available for download at [https://sites.google.com/site/gptips4matlab/].

GPTips includes 4 demos and was studied to solve the question posed in this paper. The following new scripts and files were added to the gptips directory: gpED.m, gp_configED, fitnessED. The new files are included in the appendix.

The script gpED.m is the master script to be run first which calls gd_configED.m. The script gp_configED.m contains the configuration parameters for our genetic program. These parameters are:

Population size: 100	Set termination at Error = 0.01
Number of generations: 100	Set x and y values from table
Display after 25 generations	Set non-x terminal nodes: 0,1,2,3,4,5
Elite selection fraction = 0.25	Probability of non-x terminal: 0.2
Elite number: 25	Max tree depth = 4
Point to fitnessED	Not a multigene evolution
Set fitness to select minimum error	Use functions {times, plus}

The fitnessED.m function called the GPTips function evalfitness.m which was not modified by me. The evalfitness.m function evaluates the values of the generated S-expression tree and returns an output array. The fitness of the GPTips generated tree is evaluated in my fitnessED.m script as a difference between the generated S-expression outputs and expected outputs corresponding to the table provided in the problem.

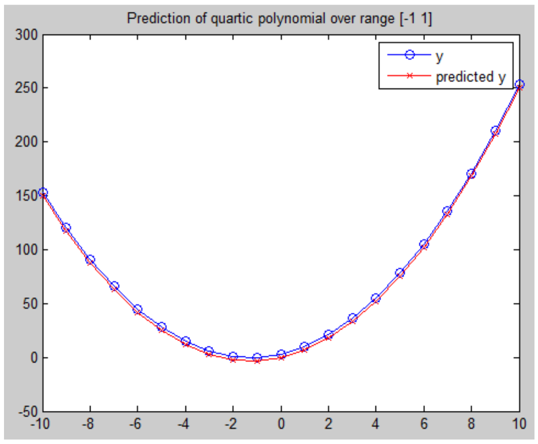
When the 100 generation had been evaluated or the termination was reached gpEd displayed the overall function of x which produced the best y in both algebraic form and S-expression forms. The predicted function was also plotted over the expected. A sample run is shown on the following page.

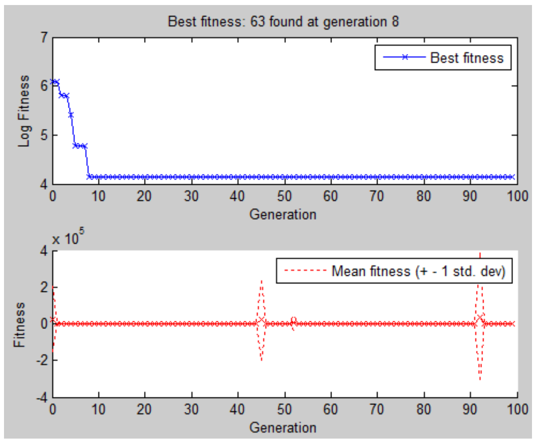
Sample run 1

```
Run parameters
--------------
Population size:        100
Number of generations:   100
Tournament size:        20
Lexicographic selection: False
Max tree depth:          4
Max nodes per tree:      Inf
Using function set:      TIMES PLUS
Number of inputs:        1
Constants range:         [0  5]
Using fitness function:  fitnessED.m

Generation 0
Best fitness:   443
Mean fitness:   25770.88
Best nodecount: 7

Generation 25
Best fitness:   63
Mean fitness:   253.92
Best nodecount: 13

Generation 50
Best fitness:   63
Mean fitness:   258.15
Best nodecount: 13

Generation 75
Best fitness:   63
Mean fitness:   530.59
Best nodecount: 13

GPTIPS run complete.
Best fitness acheived: 63
------------------------------------------------------------

Simplified overall GP expression:
---------------------------------
```

$2.0\, x_1^2 + 5.0\, x_1$

```
plus(plus(times([3],x1),plus(x1,x1)),times(plus(x1,x1),x1))
```





This run didn't reach the perfect solution after 100 generations. It came up with a function:

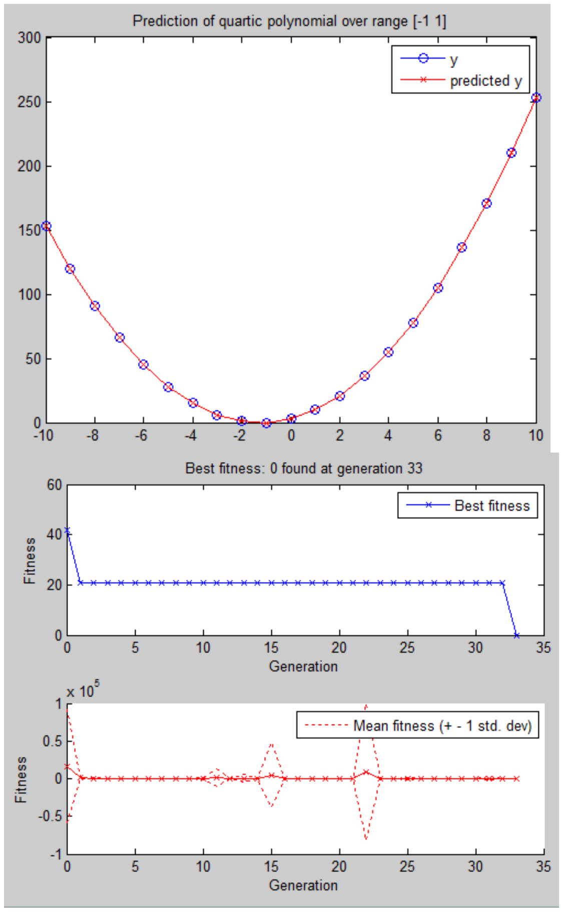
$$y = 2.0\, x^2 + 5.0\, x$$

Sample run 2

```
Run parameters
--------------
Population size:        100
Number of generations:   100
Tournament size:        20
Lexicographic selection: False
Max tree depth:         4
Max nodes per tree:      Inf
Using function set:      TIMES PLUS
Number of inputs:        1
Constants range:        [0  5]
Using fitness function:  fitnessED.m

Generation 0
Best fitness:   42
Mean fitness:   16655.7
Best nodecount: 15

Generation 25
Best fitness:   21
Mean fitness:   266.84
Best nodecount: 15

Fitness criterion met. Terminating run.
GPTIPS run complete.
Best fitness acheived: 0
---------------------------------------------------------

Simplified overall GP expression:
---------------------------------
```

$2.0\ x_1^2\ + 5.0\ x_1 + 3.0$

```
plus(plus(times(x1,[4]),times(x1,x1)),plus(plus([3],x1),times(x1,x1)))
```



This run did reach the perfect solution after less than 35 generations. It came up with a function:

$$y = 2.0\ x^2\ + 5.0\ x + 3$$
$$= 1*(2*x+3)(x+1)$$

The table below shows solutions, their fitness, and how many generations were required to reach them.

Solution	Best Fitness	Generation
1*(2*x+3)*(x+1)	0	4
1*(2*x+1)*(x+2)	21	100
$2*x^2+5*x+4$	21	100
$2*x^2+5*x+3$	0	3
$2*x^2+5*x+2$	21	100

Appendix
gpED.m

```
%Edris Amin
%GP homework for problem #7 pp117
clc

gp=rungp('gp_configED');
summary(gp);
runtree(gp,'best');

%display equation
gppretty(gp,'best');
expression  = gp.results.best.eval_individual{1};
disp(expression);
```

fitnessED.m

```
%Edris Amin
%GP fitness function
%code edited from original from (c) Dominic Searson 2009
function [fitness,gp]=fitnessED(evalstr,gp)

% Extract x and y data from GP struct
x1=gp.userdata.x;
y=gp.userdata.y;

% Evaluate the tree (assuming only 1 gene is suppled in this case - if
the
% user specified multigene config then only the first gene encountered
will be used)

eval(['out=' evalstr{1} ';']);

% Fitness is sum of absolute differences between actual and predicted y
fitness=sum(abs(out-y));

% If this is a post run call to this function then plot graphs
if gp.state.run_completed
    figure
    plot(x1,y,'o-');hold on;
    plot(x1,out,'rx-');
    legend('y','predicted y');
    title('Prediction of quartic polynomial over range [-1 1]');
    hold off;
end
```

gp_configED.m

```
%Edris Amin
function gp=gp_configED(gp);

gp.runcontrol.pop_size=100; %population size
gp.runcontrol.num_gen=100; %number of generations = 0:num_gen
gp.runcontrol.verbose=25; %after how many generations to display information

% Selection method options-------------------------
gp.selection.tournament.size=4;
% gp.selection.tournament.lex_pressure=true; % True to use Luke & Panait's
plain lexicographic tournament selection
gp.selection.elite_fraction=0.02;       %  Elitist selection fraction of
population to copy directly to next generation without modification.

% Fitness function specification------------------------------
gp.fitness.fitfun=@fitnessED; % fitness function pointer (no need of .m using
'@name')
gp.fitness.minimisation=true; % Set to true if you want to minimise the
fitness function (if false it is maximised).
gp.fitness.terminate=true;    %terminate before num_gen reached
gp.fitness.terminate_value=0.1; %terminate here if termination = true

% Set up user data-----------------
xin=linspace(-10,10,21); %generate [-10, -9, -8, ... -1, 0, 1, ... 8 , 9 ,
10]
yin=[153 120 91 66 45 28 15 6 1 0 3 10 21 36 55 78 105 136 171 210 253];
gp.userdata.x=xin;
gp.userdata.y=yin;

% Input configuration--------------------
%Define the number of inputs
gp.nodes.inputs.num_inp=1; %inputing one x value per individual

% Constant parameters---------
gp.nodes.const.num_dec_places = 0; %# of dec. places at nodes (integers = 0)
gp.nodes.const.range = [0 5]; %range of values nodes can take if not x

%Probability that a constant node, rather than an input node, will be
generated when adding a terminal node to a tree.
gp.nodes.const.p_ERC = 0.2;

% Tree build options-------------------
gp.treedef.max_depth=4; % Maximum depth of trees
% Maximum depth of sub-trees created by mutation operator
gp.treedef.max_mutate_depth=4; %same as max_depth after mutation

%Enable multiple gene mode and set max number of genes per individual.
gp.genes.multigene=false; %no multigene used

%Define function nodes
gp.nodes.functions.name{1} = 'times';
gp.nodes.functions.name{2} = 'plus';
```